

| **Title:** Solving planning problem using STRIPS or PDDL tools. |
| --- |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Objective:** To write STRIPS scripts to solve planning problem and implement them using PDDL tools

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected Outcome of Experiment:**

| **Course Outcome** | **After successful completion of the course students should be able to** |
| --- | --- |
| **CO2** | Analyze and solve problems for goal based agent architecture (searching and planning algorithms). |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. [**https://planning.wiki/**](https://planning.wiki/)**, last retrieved on Feb 27,2025**
2. [**https://editor.planning.domains/**](https://editor.planning.domains/)**, last retrieved on Feb 27,2025**
3. [**https://www.youtube.com/watch?v=EeQcCs9SnhU**](https://www.youtube.com/watch?v=EeQcCs9SnhU)**, last retrieved on Feb 27,2025**
4. [**https://www.youtube.com/watch?v=FS95UjrICy0**](https://www.youtube.com/watch?v=FS95UjrICy0)**, last retrieved on Feb 27,2025**
5. [**https://nms.kcl.ac.uk/planning/software/optic.html**](https://nms.kcl.ac.uk/planning/software/optic.html)**, last retrieved on Feb 27,2025**
6. [**https://github.com/yarox/pddl-examples**](https://github.com/yarox/pddl-examples)**, last retrieved on Feb 27,2025**
7. [**https://planning.wiki/\_citedpapers/pddl3bnf.pdf**](https://planning.wiki/_citedpapers/pddl3bnf.pdf) **, last retrieved on Feb 27,2025**
8. **https://github.com/potassco/pddl-instances, last retrieved on Feb 27,2025**
9. **“Artificial Intelligence: a Modern Approach” by Russell and Norving, Pearson education Publications**
10. **“Artificial Intelligence” By Rich and knight, Tata McGraw Hill Publications**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Pre Lab/ Prior Concepts:**

Goal based agents, searching, uninformed search, informed search

**Historical Profile:** *(Details about planning Vs Searching)*

#### 1. Searching:

* **Definition:** Searching involves finding a path from an initial state to a goal state using a sequence of actions.
* **Types of Search:**
  + **Uninformed Search:** No domain knowledge, examples include BFS, DFS, and Dijkstra’s algorithm.
  + **Informed Search:** Utilizes heuristic knowledge, examples include A\* and Greedy Best-First Search.
* **Limitation:** Searching techniques explore a potentially large state space, making them inefficient in complex domains with many actions.

#### 2. Planning:

* **Definition:** Planning involves determining a sequence of actions that leads from an initial state to a goal state, taking into account the effects and constraints of actions.
* **Advantages over Searching:**
  + Planning uses **high-level abstractions** and **logical representations** to reduce state space complexity.
  + It explicitly models the actions, preconditions, and effects, ensuring that actions contribute towards achieving the goal.
* **Categories of Planning:**
  + **Total Order Planning (TOP):** Generates a linear sequence of actions.
  + **Partial Order Planning (POP):** Maintains a flexible plan where actions are ordered only when necessary.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**New Concepts to be learned:**

#### 1. Representing Problems as Planning Problems

* Convert a real-world problem into a **planning problem** by defining:
  + **Initial State:** A description of the starting situation.
  + **Goal State:** Desired outcome that needs to be achieved.
  + **Actions:** Defined by preconditions and effects.

#### 2. STRIPS (Stanford Research Institute Problem Solver)

* STRIPS formalizes the representation of actions, states, and goals in a planning problem.
* **Components:**
  + **Initial State:** List of facts that describe the world at the beginning.
  + **Goal State:** Set of conditions that must hold true to achieve the goal.
  + **Actions:** Defined as:
    - Preconditions: Conditions that must hold before an action.
    - Add List: Facts added to the state after action execution.
    - Delete List: Facts removed from the state after action execution.

#### 3. ADL (Action Description Language)

* ADL extends STRIPS by allowing:
  + Disjunctive preconditions
  + Conditional effects
  + Existential and universal quantification
  + Negative preconditions

#### 4. Total Order Plan (TOP)

* Generates a strictly ordered sequence of actions from the initial state to the goal.
* All actions are executed in a predefined sequence.
* Suitable for domains where actions need strict ordering.

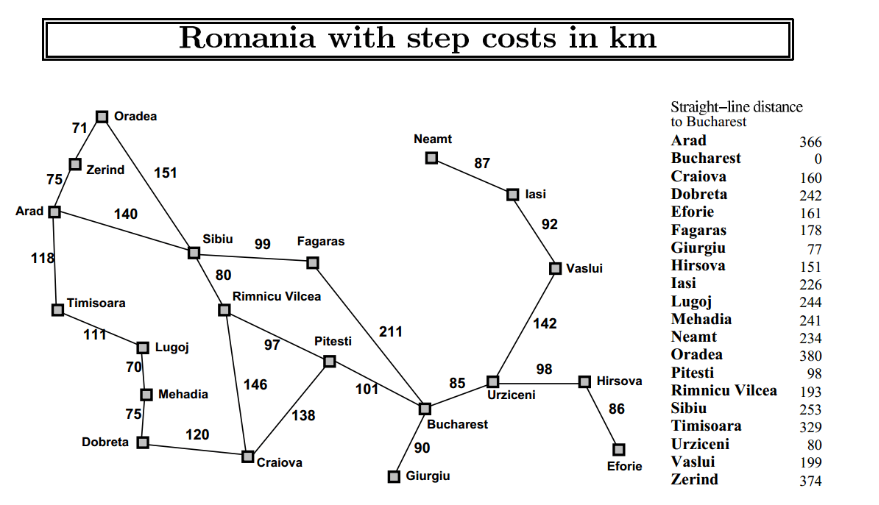
#### 5. Partial Order Plan (POP)

* Constructs a plan where actions are partially ordered based on necessity.
* Provides flexibility by allowing concurrency between independent actions.
* POP resolves conflicts dynamically using causal links.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Chosen Planning Problem:**

The objective is to model and solve a transportation problem for the Map of Romania, where an agent (a traveler or a vehicle) must navigate the country's road network to reach a specified goal location — Bucharest.The domain is structured to represent different cities as locations and roads as connections between them. The agent can "drive" between connected cities, ensuring logical consistency in movement.

****

**Initial States:** Arad , Sibiu , Timișoara , Vaslui , Efforie  **Goal State:** Bucharest

**PDDL Script for solving problem:**

**Domain:**(define (domain romania)

(:requirements :strips :typing)

(:types city)

(:predicates

(at ?c - city)

(road ?from ?to - city)

)

(:action drive

:parameters (?from ?to - city)

:precondition (and (at ?from) (road ?from ?to))

:effect (and (at ?to) (not (at ?from))))

)

**Problem:**

**Problem 1: Starting from Arad**

(define (problem romania-p1)

(:domain romania)

(:objects

Arad Bucharest Sibiu Fagaras RimnicuVilcea Craiova Drobeta Mehadia Lugoj Oradea Zerind Pitesti Timisoara - city)

(:init

(at Arad)

(road Arad Zerind)

(road Zerind Oradea)

(road Oradea Sibiu)

(road Sibiu Fagaras)

(road Fagaras Bucharest)

(road Sibiu RimnicuVilcea)

(road RimnicuVilcea Craiova)

(road Craiova Pitesti)

(road Pitesti Bucharest)

(road Timisoara Lugoj)

(road Lugoj Mehadia)

(road Mehadia Drobeta)

(road Drobeta Craiova)

)

(:goal (at Bucharest))

)

**Problem 2: Starting from Sibiu**

(define (problem romania-p2)

(:domain romania)

(:objects

Arad Bucharest Sibiu Fagaras RimnicuVilcea Craiova Drobeta Mehadia Lugoj Oradea Zerind Pitesti Timisoara - city)

(:init

(at Sibiu)

(road Arad Zerind)

(road Zerind Oradea)

(road Oradea Sibiu)

(road Sibiu Fagaras)

(road Fagaras Bucharest)

(road Sibiu RimnicuVilcea)

(road RimnicuVilcea Craiova)

(road Craiova Pitesti)

(road Pitesti Bucharest)

(road Timisoara Lugoj)

(road Lugoj Mehadia)

(road Mehadia Drobeta)

(road Drobeta Craiova)

)

(:goal (at Bucharest))

)

**Problem 3: Starting from Timisoara**

(define (problem romania-p5)

(:domain romania)

(:objects

Arad Bucharest Sibiu Fagaras RimnicuVilcea Craiova Drobeta Mehadia Lugoj Oradea Zerind Pitesti Timisoara - city)

(:init

(at Timisoara)

(road Arad Zerind)

(road Zerind Oradea)

(road Oradea Sibiu)

(road Sibiu Fagaras)

(road Fagaras Bucharest)

(road Sibiu RimnicuVilcea)

(road RimnicuVilcea Craiova)

(road Craiova Pitesti)

(road Pitesti Bucharest)

(road Timisoara Lugoj)

(road Lugoj Mehadia)

(road Mehadia Drobeta)

(road Drobeta Craiova)

)

(:goal (at Bucharest))

)

**Problem 4: Starting from Vaslui**

(define (problem romania-p6)

(:domain romania)

(:objects

Arad Bucharest Sibiu Fagaras RimnicuVilcea Craiova Drobeta Mehadia Lugoj Oradea Zerind Pitesti Timisoara

Vaslui Urziceni Hirsova Eforie - city)

(:init

(at Vaslui)

;; Northern/Central Branch

(road Arad Zerind)

(road Zerind Oradea)

(road Oradea Sibiu)

(road Sibiu Fagaras)

(road Fagaras Bucharest)

(road Sibiu RimnicuVilcea)

(road RimnicuVilcea Craiova)

(road Craiova Pitesti)

(road Pitesti Bucharest)

;; Southern Branch

(road Timisoara Lugoj)

(road Lugoj Mehadia)

(road Mehadia Drobeta)

(road Drobeta Craiova)

;; East Branch

(road Vaslui Urziceni)

(road Urziceni Bucharest)

(road Eforie Hirsova)

(road Hirsova Urziceni)

)

(:goal (at Bucharest))

)

**Problem 5: Starting from Eforie**

(define (problem romania-p7)

(:domain romania)

(:objects

Arad Bucharest Sibiu Fagaras RimnicuVilcea Craiova Drobeta Mehadia Lugoj Oradea Zerind Pitesti Timisoara

Vaslui Urziceni Hirsova Eforie - city)

(:init

(at Eforie)

;; Northern/Central Branch

(road Arad Zerind)

(road Zerind Oradea)

(road Oradea Sibiu)

(road Sibiu Fagaras)

(road Fagaras Bucharest)

(road Sibiu RimnicuVilcea)

(road RimnicuVilcea Craiova)

(road Craiova Pitesti)

(road Pitesti Bucharest)

;; Southern Branch

(road Timisoara Lugoj)

(road Lugoj Mehadia)

(road Mehadia Drobeta)

(road Drobeta Craiova)

;; Eastern Branch

(road Vaslui Urziceni)

(road Urziceni Bucharest)

(road Eforie Hirsova)

(road Hirsova Urziceni)

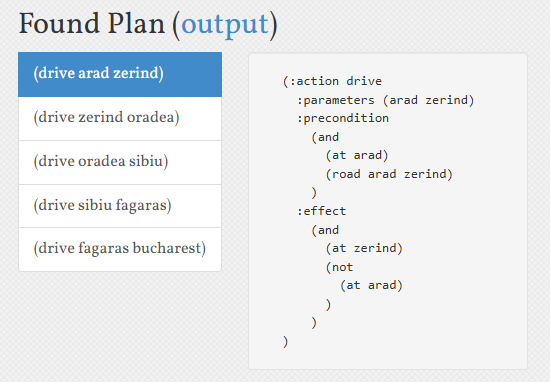
)

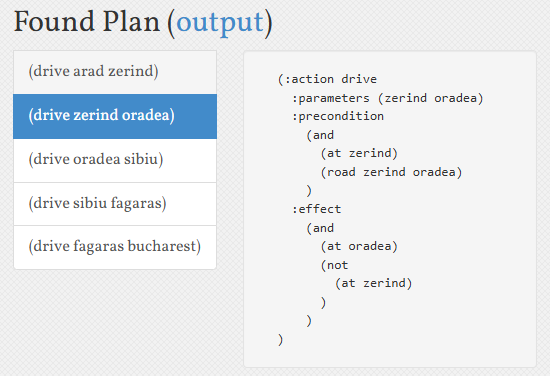
(:goal (at Bucharest))

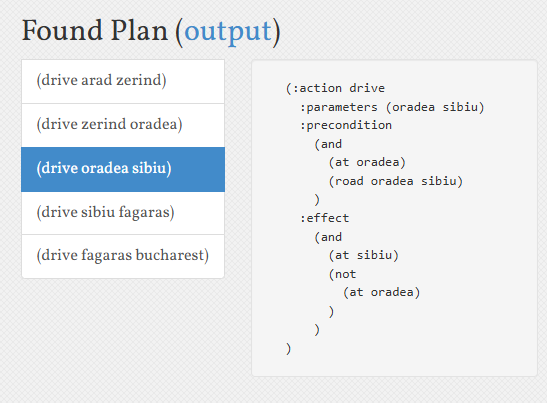
)

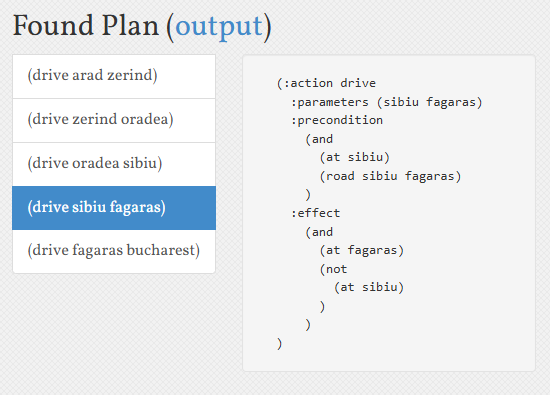
**OUTPUT:**

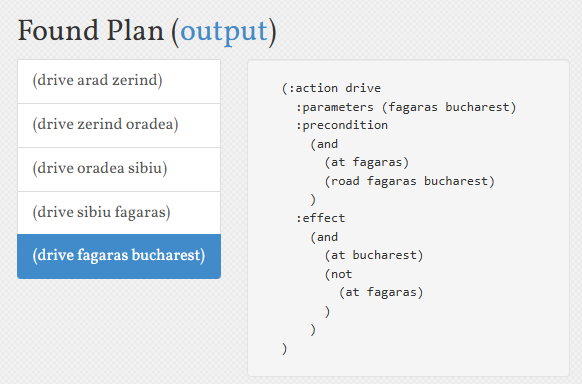
**Problem 1: Arad to Bucharest:**

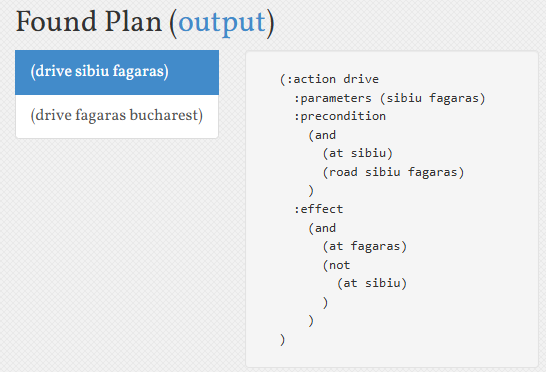
****

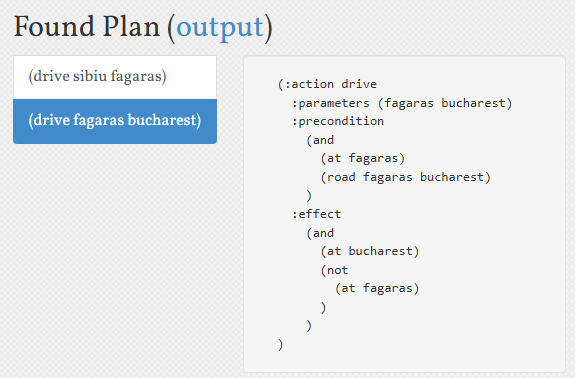
****

****

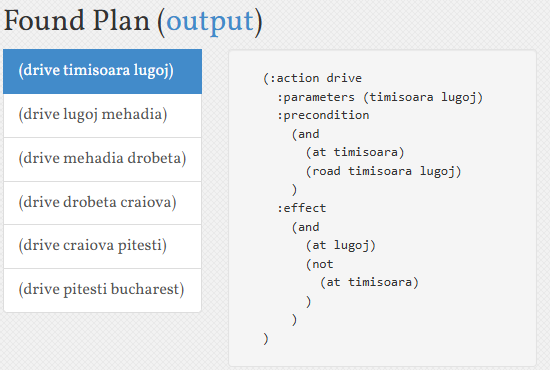
****

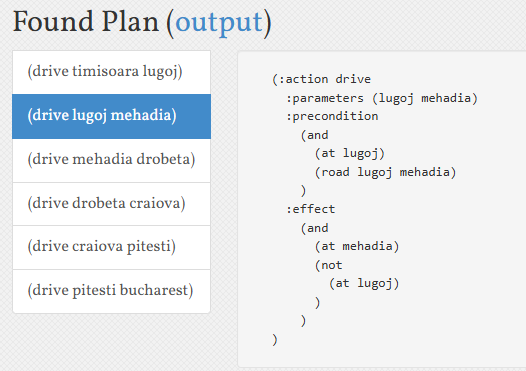
****

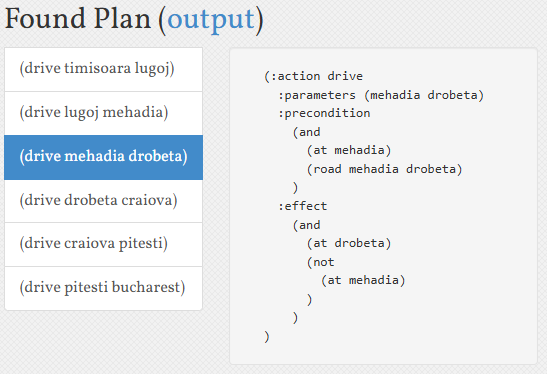
**Problem 2: Sibiu to Bucharest  
**

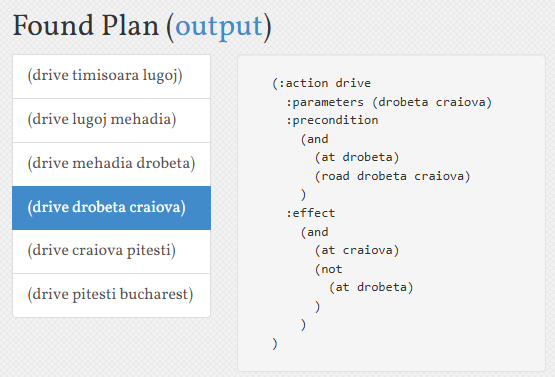
****

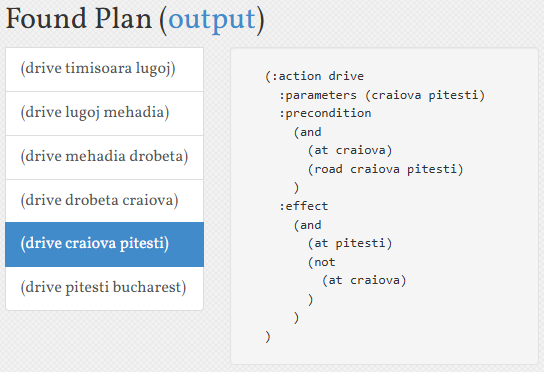
**Problem 3: Timisoara to Bucharest**

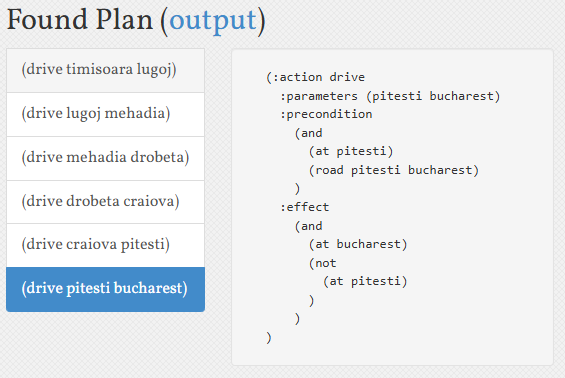
****

****

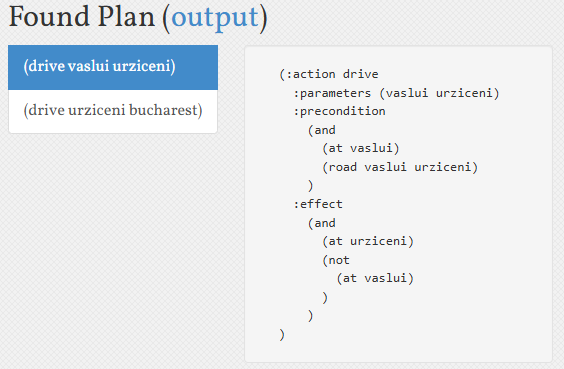
****

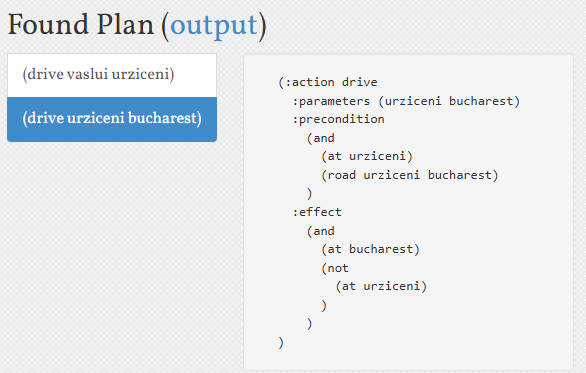
****

****

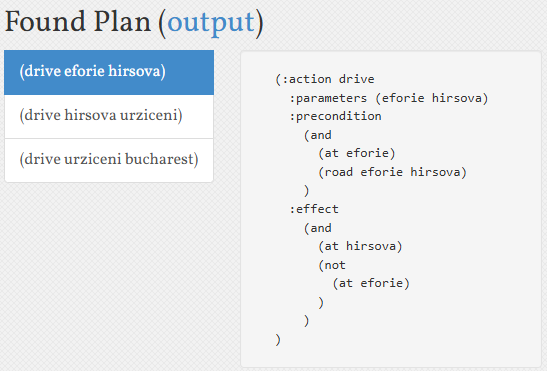
****

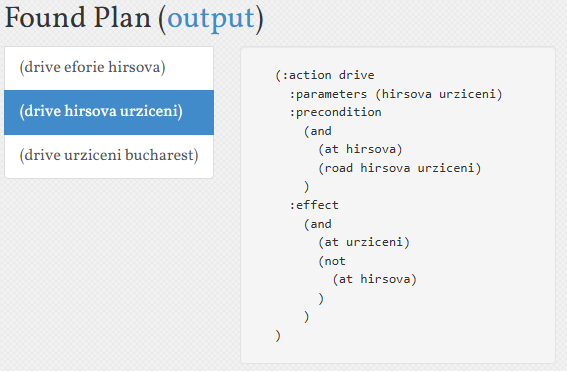
**Problem 4: Vaslui to Bucharest**

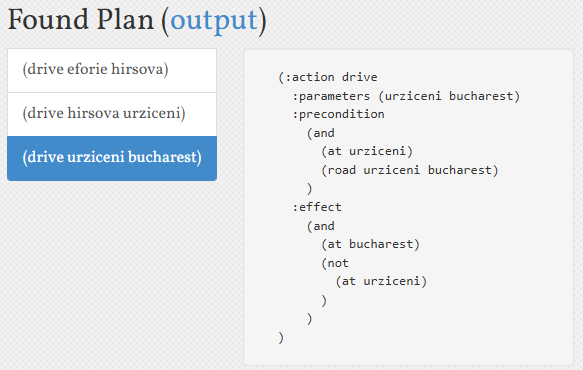
****

****

**Problem 5: Eforie to Bucharest**

****

****

****

**Google Drive Link for the PDDL files:**[**https://drive.google.com/drive/folders/1vYuekGEVoc1xypZE0c94WL133mEbtGsc?usp=sharing**](https://drive.google.com/drive/folders/1vYuekGEVoc1xypZE0c94WL133mEbtGsc?usp=sharing)

**Explanation of PDDL Model:**

* **Domain Definition:** The model is written in PDDL using STRIPS-like constructs. In the domain, we define:  
  + **Types:** Cities (or nodes) that represent locations.
  + **Predicates:** For example, (at ?c) denotes the current location of the agent, and (road ?from ?to) represents a directed connection between two cities.
  + **Actions:** The primary action is **drive**:

(:action drive

:parameters (?from ?to - city)

:precondition (and (at ?from) (road ?from ?to))

:effect (and (at ?to) (not (at ?from))))

This action is applicable when the agent is at a city that has a road leading to another city. Executing the action moves the agent from one city to another by removing the current location predicate and adding the new one.

* **Problem Definition:**Multiple problem instances are created by varying the initial city of the agent while the goal remains constant—reaching Bucharest. Each problem instance includes:
  + **Objects:** A list of cities.
  + **Initial State:** Specifies the starting city (e.g., Arad, Sibiu, Lugoj, Oradea, Timisoara, Vaslui, or Eforie) along with the road network.
  + **Goal State:** The goal is to have the predicate (at Bucharest) hold true.

### 

### 

### About the LAMA-first Solver

* **Solver Characteristics:** LAMA-first is a satisficing planner designed for speed rather than optimality. It quickly generates a plan by:  
  + **Greedy Best-First Search:** The solver uses a heuristic that favors moves leading toward the goal, but without refining or improving the plan cost after an initial solution is found.
  + **No Subsequent Refinements:** Unlike the standard LAMA planner, which might perform further iterations to optimize the plan, LAMA-first stops after the first acceptable plan is found. This makes it useful when a quick solution is more important than the absolute quality of the plan.
* **Implications for the PDDL Model:**
  + **Model Simplicity:** Since the domain (the map of Romania) is relatively straightforward—moving from city to city along defined roads—a greedy best-first search is often sufficient to find a valid route to Bucharest.
  + **Plan Cost Consideration:** Because LAMA-first does not improve upon the initial solution, the generated plan may not be the shortest or least-cost path, but it will be found quickly. This trade-off is acceptable in many applications where a valid solution is preferred over an optimal one.

**Post Lab Descriptive Questions:**

1. **How does ADL (Action Description Language) extend STRIPS?**

ADL (Action Description Language) extends **STRIPS (Stanford Research Institute Problem Solver)** by introducing more expressive representations for actions, goals, and states. The extensions include:

* **Disjunctive Preconditions:** ADL allows disjunctions (OR conditions) in action preconditions, whereas STRIPS only allows conjunctions.
* **Conditional Effects:** ADL enables effects to be conditionally applied based on the state, whereas STRIPS only allows unconditional effects.
* **Existential and Universal Quantification:** ADL supports quantified variables in preconditions and effects, which allows more general and flexible rules.
* **Negative Preconditions:** ADL allows explicitly stating what must *not* be true, unlike STRIPS which assumes closed-world conditions.

1. Define **Partial Order Planning (POP)** and **Total Order Planning (TOP)**. How do they differ?

* **Partial Order Planning (POP):**
  + Constructs a plan where the order of actions is only partially specified.
  + Actions are added as needed, and ordering constraints are imposed only when necessary to resolve conflicts or maintain causality.
  + Allows flexibility in executing actions, minimizing unnecessary constraints.
* **Total Order Planning (TOP):**
  + Constructs a linear sequence of actions where all actions are ordered from the start to the goal.
  + Requires a strict sequence, which may introduce unnecessary constraints and limit flexibility.

**Difference:**

* POP defers action ordering until absolutely necessary, promoting flexibility and reducing unnecessary constraints.
* TOP enforces a rigid order from the start, making it less efficient when multiple actions can be executed concurrently.

1. Would **Partial Order Planning** be beneficial in this problem? Why or why not?

Partial Order Planning (POP) would be beneficial if the problem involves:

* **Concurrency Potential:** If multiple actions can be performed in parallel without conflict, POP can reduce constraints and improve efficiency.
* **Complex Causal Dependencies:** When many actions have dependencies that can be resolved independently, POP avoids unnecessary sequencing.
* **Flexibility in Execution:** POP allows different execution orders to achieve the same goal, which is useful when the environment is dynamic or uncertain.

If the problem requires strict sequential execution or has very few independent actions, POP might not offer significant benefits.

1. **Describe a situation where causal links would be needed in Partial Order Planning for this problem.**

A **causal link** is needed in POP when an action provides a precondition for another action, ensuring that no intermediate action invalidates that condition.

**Example Situation:**

* **Task:** Building a beer shipment order where supplies are assembled and then shipped.
* **Step 1:** Action A: Assemble raw materials (raw\_materials\_ready becomes true).
* **Step 2:** Action B: Package the beer (raw\_materials\_ready is required).
* **Step 3:** Action C: Ship the packaged order.

A causal link between A and B would guarantee that raw\_materials\_ready remains true until B is executed. If an intermediate action threatens to negate this condition, POP would introduce ordering constraints to prevent interference.